

AMENDMENTS TO THE CLAIMS

1 1. (Original) A method for accessing a unified memory in a micro-processing system
2 having a microprocessor, a one level pipeline, and a two-phase clock, such that all instructions are
3 executed in a single cycle, comprising:

4 (a) fetching a program instruction from the unified memory;

5 (b) determining if the fetched program instruction would require three unified memory
6 accesses during a single instruction cycle for proper execution of the fetched program instruction,
7 proper execution of the fetched program instruction being the microprocessor performing the
8 operations requested by the fetched program instruction in a single instruction cycle;

9 (c) accessing the unified memory a first time, during the instruction cycle associated with
10 the fetched program instruction, with a dummy access when it is determined that the fetched
11 program instruction requires three unified memory accesses for proper execution of the fetched
12 program instruction;

13 (d) fetching a next program instruction from an instruction register, during the instruction
14 cycle associated with the fetched program instruction, when it is determined that the fetched
15 program instruction requires three unified memory accesses for proper execution of the fetched
16 program instruction; and

17 (e) accessing the unified memory a second time, during the instruction cycle associated
18 with the fetched program instruction, with a data access when it is determined that the fetched
19 program instruction requires three unified memory accesses for proper execution of the fetched
20 program instruction.

1 2. (Original) The method as claimed in claim 1, wherein the data access is a read data

2 access.

1 3. (Original) The method as claimed in claim 1, wherein the data access is a write data
2 access.

1 4. (Original) The method as claimed in claim 1, wherein the fetched program instruction
2 is a last instruction of a loop.

1 5. (Original) The method as claimed in claim 1, wherein the instruction register is an
2 instruction stack, thereby enabling program instruction fetches for nested loops.

1 6. (Original) A method for accessing a unified memory in a micro-processing system
2 having a microprocessor, a one level pipeline, and a two-phase clock, such that all instructions are
3 executed in a single cycle, comprising:

4 (a) fetching a program instruction from the unified memory during a first instruction cycle;

5 (b) determining if the fetched program instruction for a second instruction cycle is a
6 conditional program code discontinuity;

7 (c) accessing the unified memory a first time during the second instruction cycle with a
8 dummy access when it is determined that the program instruction accessed for a second instruction
9 cycle is a conditional program code discontinuity; and

10 (d) accessing the unified memory a second time during the second instruction cycle to read
11 a new instruction when it is determined the program instruction accessed for a second instruction
12 cycle is a conditional program code discontinuity, thereby delaying the instruction access from the
13 unified memory for the second instruction cycle by a half cycle.

1 7. (Original) The method as claimed in claim 6, wherein the conditional program code

2 discontinuity is a jump instruction.

1 8. (Original) The method as claimed in claim 6, wherein the conditional program code
2 discontinuity is a call instruction.

1 9. (Original) A method for accessing a unified memory in a micro-processing system
2 having a microprocessor, a one level pipeline, and a two-phase clock, such that all instructions are
3 executed in a single cycle, comprising:

4 (a) fetching a program instruction from the unified memory;

5 (b) determining if the fetched program instruction is a loop initiation instruction;

6 (c) storing a first instruction of the loop in an instruction register when the fetched
7 program instruction is a loop initiation instruction;

8 (d) executing the loop;

9 (e) determining if a fetched instruction during the execution of the loop is a last instruction
10 of the loop;

11 (f) accessing the unified memory a first time, during the instruction cycle associated with
12 the fetched last instruction of loop, with a dummy access;

13 (d) fetching the first instruction of the loop from the instruction register, during the
14 instruction cycle associated with the fetched last instruction of loop; and

15 (e) accessing the unified memory a second time, during the instruction cycle associated
16 with the fetched last instruction of loop, with a data access.

1 10. (Original) The method as claimed in claim 9, wherein the data access is a read data
2 access.

1 11. (Previously Presented) The method as claimed in claim 9, wherein the data access is a
2 write data access.

1 12. (Previously Presented) The method as claimed in claim 9, wherein the instruction
2 register is an instruction stack, thereby enabling program instruction fetches for nested loops.

1 13. (Previously Presented) A method for accessing a unified memory in a micro-
2 processing system during a loop instruction, comprising:

3 (a) accessing a program instruction from the unified memory during a first instruction
4 cycle;

5 (b) determining a type of program instruction;

6 (c) pre-fetching a next instruction from the unified memory;

7 (d) saving the pre-fetched instruction in a register when it is determined that the type of
8 program instruction is a first instruction of a loop;

9 (e) fetching a next instruction from the register when it is determined that the type of
10 program instruction is a last instruction of a loop;

11 (f) accessing the unified memory with a dummy access during execution of the last
12 instruction of the loop; and

13 (g) accessing the unified memory, a second time, with a data access during execution of the
14 last instruction of the loop.

1 14. (Currently Amended) The method as claimed in claim ~~{14}~~ 13, wherein the method
2 saves the pre-fetched instruction in a stack register when it is determined that the type of program
3 instruction is first instruction of a loop to enable nested loops and interruptible loops, and the method

- 4 fetches a next instruction from the stack register when it is determined that the type of program
- 5 instruction is a last instruction of the loop.